

6-14-00

A

Express Mail Label Number:
EL046274071US

UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR § 1.53(b))
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PTO
09/591926
06/12/00

Commissioner of Patents and Trademarks
Box Patent Application
Washington, DC 20231

☒ Duplicate for fee processing

Sir: This is a request for filing a patent application under 37 CFR § 1.53(b) in the name of inventor(s):

Carol A. Lavelle, Joyce Z. Yu and Eric G. Sultan

Title: **DEVICE POWER MANAGEMENT**

Application Elements:

- ☒ **21** Pages of Specification, Claims and Abstract
- ☒ **04** Sheets of Drawings
- ☒ **03** Pages Combined Declaration and Power of Attorney

Accompanying Application Parts:

- ☐ Assignment and Assignment Recordation Cover Sheet (recording fee of \$40.00 enclosed)
- ☒ Information Disclosure Statement with Form PTO-1449
 - ☒ Copies of IDS Citations
- ☐ Preliminary Amendment
- ☒ Return Receipt Postcard
- ☐ Other: **N/A**


Fee Calculations (37 CFR § 1.16):

	<u>No. Filed</u>		<u>No. Extra</u>	<u>Rate</u>	<u>Fee</u>
Basic Fee					\$ 690.00
Total Claims	29	-20 =	9	x \$18 =	\$ 162.00
Independent Claims	8	-03 =	5	x \$78 =	\$ 390.00
<input type="checkbox"/> Multiple Dependent Claim Presented				\$260=	\$ 0.00
TOTAL FILING FEE:					\$1242.00

- ☒ Please charge Deposit Account no. **19-4974** in the amount of **\$1,242.00** to cover the total filing fees.
- ☒ The Commissioner is hereby authorized to charge any additional fees which may be required, including extension fees, or credit any overpayment to Deposit Account No. **19-4974**.
- ☒ Please send correspondence to the following address:

SUN MICROSYSTEMS
901 San Antonio Road MS UPAL01-521
Palo Alto, CA 94303

Attention of Bernice B. Chen Phone: (650) 336-0897; Fax (650) 336-0530.

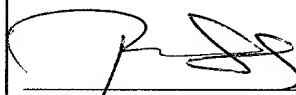
Date: 6-12-00 
Bernice B. Chen
 Registration No. **42,403**

CERTIFICATE OF EXPRESS MAIL
 UNDER 37 C.F.R § 1.10

"Express Mail" mailing label number: **EL046274071US**

DATE OF DEPOSIT: June 12, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. §1.10 on the date indicated above and is addressed to: Commissioner of Patents
 Box Patent Application
 Washington DC 20231


 Bernice B. Chen

00250-3360530

Express Mail Label No.:

EL046274071US

UNITED STATES PATENT APPLICATION
FOR

DEVICE POWER MANAGEMENT

INVENTORS:

CAROL A. LAVELLE
JOYCE Z. YU
ERIC G. SULTAN

FIELD OF THE INVENTION

This invention relates to power management of devices in a computer system, in particular, to power management of computer peripheral devices.

BACKGROUND OF THE INVENTION

Power management is desirable for desktop personal computers and workstations because a typical desktop computer system can consume several hundred watts of power per hour when it is turned on, whether or not it is being used. This energy consumption can be quite expensive, particularly in the context of large companies and other institutions that may have hundreds or even thousands of desktop computer systems turned on both day and night. One way to reduce power consumption is to power manage the computer system by, for example, turning off peripheral devices that are not in use or slow down the processes that are running on the computer.

Energy Star (EStar) guidelines are power management guidelines issued by the U.S. Government's Environmental Protection Agency (EPA). EStar guidelines exist for many products, including desktop computers. EStar power management guidelines require specified reduction in power within a specified time for a computer system when no keyboard or mouse activity has been detected. For example, the proposed Memorandum of Understanding 3 (MOU-3) guidelines require the display framebuffer to be powered off after 30 minutes when no keyboard or mouse activities have been detected.

Powering down the framebuffer during power management may present problems to systems that are running window applications that do not require keyboard or mouse activities. For example, a clock tool may continuously update the time display in a window thus requiring access to the display framebuffer regardless of whether there are user activities at the keyboard or the mouse. If the computer system

where the window is displayed enters power management mode while the clock tool is running, the clock tool continues to access the framebuffer even though the framebuffer is powered off. Access to the framebuffer when it is powered off causes the system to hang.

5

SUMMARY OF THE INVENTION

The invention relates to methods and apparatus for allowing a window system that is running processes that access a device but do not require keyboard or mouse interactions to continue running while the device is powered off.

10

In accordance with the invention, access to the device is prevented when the computer system is in power management state by directing data intended for the device to a memory location. In one embodiment, device registers and memory are accessed via a first mapping supplied by a device driver. When the system is going into power management state, the first mapping is modified to a second mapping. A memory location is allocated for the second mapping which substitutes for the first mapping prior to the device being physically powered off. This second mapping directs all device access to the allocated memory during power management state. When the system exits power management state, the first mapping is restored and the device registers and memory are updated.

15

In one embodiment, when an idle state has been detected, a device-independent program in an X Server makes a call to a device-dependent program to request power management of a device. The device-dependent program makes a system call to a device driver (e.g., a framebuffer device driver) which then allocates a portion of host memory and informs the device-dependent program to map data intended for the framebuffer to the allocated memory. The device-dependent program requests the power manager to power off the framebuffer. Power manager informs the device driver of the pending framebuffer power off prior to powering off the framebuffer so

20

that the device driver may perform pre-power-down tasks such as saving device state. After power down, data intended for the framebuffer is directed to the allocated portion of the host memory.

When power is restored to the framebuffer, the content of the framebuffer is updated. In one embodiment, the power-up sequence includes the device-independent program in the X Client makes a call to the device-dependent layer requesting power to be restored to the framebuffer. The device-dependent layer which then makes a system call to the power manager to restore power to the framebuffer. The power manager restores power to the framebuffer and informs the device driver. The device driver restores device state and informs the device-dependent program. The device-dependent program makes a system call to the device driver to restore mapping to the framebuffer. The device driver releases the allocated portion of the host memory and directs mapping to the framebuffer. The device-dependent program then updates the framebuffer to its current state.

This summary is not intended to limit the scope of the invention, which is defined solely by the claims attached hereto.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an X Window System with its hardware and software components.

FIG. 2 is a conceptual illustration of software and hardware system components and relations between the components.

FIG. 3 shows a flowchart of a power management process in accordance with one embodiment of the invention.

FIG. 4 illustrates an embodiment of the management and translation of the virtual view of memory (the address space) to physical memory by a memory management unit (MMU).

While the specific embodiments are described and illustrated herein, these embodiments are not intended to limit the scope of the invention, which is susceptible to various modifications and alternative forms.

5 DETAILED DESCRIPTION OF THE INVENTION

Methods and apparatus for managing power supplied to a device in a computer system are provided. Access to the device may be prevented when the computer system is in power management state and the device is powered off. Access requests intended for the device may be redirected to an allocated portion of system memory.

10 While the system of the present invention applies to computer systems in general, the system will be described in the environment of the UNIX operating system in order to simplify the discussion.

FIG. 1 illustrates an X Window system 100 with its hardware and software components. X Window system 100 may be a windowing system that runs under UNIX or other operating systems such as DOS or Windows. X Window system 100 allows users (e.g., clients) to run applications on other computers in the network (e.g., servers) and view the output on their own displays. In other words, a remote computer may update an existing screen or display a new screen on a user machine without input from the local user (e.g., input via a mouse, keyboard, stylus, or other input devices).

X Window system 100 may include a client system 200 (e.g., client user's machine) and a server system 300 (e.g., a remote server machine). X Window system 100 may include multiple client systems and multiple server systems.

25 Client system 200 (also called an X Server) is the receiving computer in X Window system 100 and may be any suitable computer system such as a desktop, a personal computer or a workstation. In one embodiment, client system 200 includes a display screen 202, a keyboard 204, a mouse 206 and a hard drive 207.

Client system 200 runs an operating system (OS) 210, X Window Server software 220, and local client applications 230. OS 210 provides control for, for example, user interface, job management, task management, data management, and device management for client system 200. OS 210 may be a single user operating system intended for a desktop system and may be, but not limited to, UNIX, Macintosh System 7, DOS, or Windows.

X Window Server software 220 handles, e.g., communications between client applications (e.g., mail, clock, calculator), the display hardware (e.g., monitor, framebuffer), and input devices (e.g., mouse, keyboard). For example, X Window Server software 220 may handle drawings on user display 202, interfaces with device drivers to receive/transmit data from input devices such as keyboard 204 and mouse 206 and other input/output devices such as video display, speaker, scanner, small computer system interface (SCSI), etc., and manages off-screen memory, fonts, cursors, colormaps, etc. X Window Server software 220 may be X Server software for the Solaris™ operating environment by Sun Microsystems, Inc. of Palo Alto, California, or any other conventional or suitable X Window Server software.

Local client applications 230 may include applications that do not require keyboard or mouse interactions to continue running and may be, e.g., clock tool, calendar tool, mail tool, etc.

Server system 300 (also called an X Client or an X Window System Client) is the processing computer in X Window system 100 and runs, for example, a network operating system 310, X Window Client software 320 and remote client applications 330. Both X Window Server software 220 and X Window Client software 320 may be run in the same machine. However, for convenience in this description, X Window Server software 220 and X Window Client software 320 will reside on different machines for ease of explanation.

Network OS 310 manages multiple user requests at the same time and may be,

but is not limited to, Windows NT, UNIX, or Linux. Server system 300 may be any suitable servers, including Sun Enterprise™ Servers by Sun Microsystems, Inc. of Palo Alto, California. Remote client applications 330 may be similar to local client applications 230 and may be any user application programs such as a word processor, spreadsheet, database program, file transfer program, e-mail program, clock program, calculator program, etc.

Within X Window system 100 is typically a socket based protocol (e.g., TCP/IP) for transmitting graphical information between X Window Client software 320 and X Window Server software 220. For example, X Window Client software 320 and X Window Server software 220 communicate through messages, called “requests” if they travel from X Window Client software 320 to X Window Server software 220, and “events” if the messages travel from X Window Server software 220 to X Window Client software 320. X Window Server software 220 maintains a windowing user interface environment on client system 200, and X Window Client software 320 generates requests to create windows, draw objects etc. within that environment. In turn, X Window Server software 220 sends events to X Window Client software 320. Events may be packets of information informing X Window Client software 320 what the user and X Window Server software 220 have performed/executed regarding an application program (e.g. remote client applications 330). For instance, a mouse click from mouse 206 causes an event from X Window Server software 220 to be sent to X Window Client software 320, where remote client applications 330 decide what to do with the mouse event.

FIG. 2 shows a conceptual diagram of software and hardware components of an X Window System 100. For purpose of illustration, FIG. 2 illustrates system components related to power management of a display framebuffer 208. X Client 222 may be either local (e.g., at the user machine) or remote (e.g., at a server machine). X Client 222 runs applications (e.g., local client applications 230 or remote client

applications 330) that may involve, for example, graphics-based user interfaces that incorporate icons, pull-down menus and a mouse. The graphics-based user interfaces can be any suitable graphical user interface, e.g., Windows, Macintosh, and X. In a client/server environment, X Client 222 typically resides in the user's client machine, e.g., client system 200, but may reside remotely in a server machine, e.g., server system 300.

X Server (X Window Server software) 220 includes a device-independent layer (DIX) 224 and a device-dependent layer (DDX) 226. DIX 224 is a software program that works with a variety of peripheral devices while hardware-specific instructions are in some other software programs. Thus, DIX 224 typically contains functions that do not depend on graphics hardware. DIX 224 may dispatch client requests, manage the event queue, distribute events to clients, manage visible data structures, and perform other functions. X Client 222 typically makes system calls into a device independent layer (e.g., DIX 224) which then redirect the system calls to the device dependent layer (e.g., DDX 226).

DDX 226 is a software program that addresses specific hardware features and typically works with a particular type of peripheral device. DDX 226 creates and manipulates, for example, pixmaps, clipping regions, colormaps, screens, fonts, and graphic contexts. DDX 226 may contain routines that depend on graphics hardware and input devices that X Server 220 accommodates.

OS 210 includes, for example, a power manager 212 and device drivers 214. Power manager 212 manages power consumption of a computer system (e.g., client system 200 in FIG. 1). In general, power manager 212 contains a set of instructions that provide a software interface to the available power controlling hardware devices and provide routines to device drivers to control power to those hardware devices. Power manager 212 may be implemented in a power management hardware such as a microprocessor or other suitable components.

Device drivers 214 are program routines that link peripheral devices (e.g., hardware framebuffer 208) to the operating system 210 and contain the precise machine language necessary to control the devices as requested by an application. Essentially, device drivers 214 convert the more general input/output instructions of the operating system 210 into messages that devices can understand. Device drivers 214 may include, for example, framebuffer device drivers, keyboard device drivers, floppy disk device drivers, non-SCSI hard disk device drivers, printer device drivers, monitor device drivers, CD-ROM reader device drivers, etc.

In one embodiment, virtual memory mapping may be utilized. Virtual memory systems allow processes to see linear ranges of bytes in their address space regardless of the physical layout or fragmentation of the real physical memory. Virtual memory systems also allow the implementation of a programming model with a larger memory size than available physical storage (e.g., RAM) and to use slower but larger secondary storage (e.g., disk) as a backing store to hold the pieces of memory that do not fit in physical memory. In virtual memory systems, rather than managing every byte of memory, page-sized pieces of memory are used to minimize the amount of work the virtual memory system has to do to maintain virtual to physical memory mapping.

FIG. 4 illustrates an embodiment of virtual memory mapping and shows how the management and translation of the virtual view of memory (the address space) to physical memory is performed by a hardware unit, known as the virtual memory management unit (MMU). In this embodiment, the kernel uses paged memory management which relies on MMU 530 to translate the virtual addresses into physical addresses.

An X Server has an address space 500 that has addresses starting from 0000. Address space 500 includes program codes, stacks and so forth. The kernel breaks up address space 500 into segments, e.g., one for each type of memory area in the address space. For example, a simple process has a memory segment 510 for the process

binary 504 and a memory segment 512 for the scratch memory 502 (known as heap space). Each segment manages the mapping for the page-sized virtual address range 520 mapped by that segment and converts that mapping into physical memory pages 540 by implementing a lookup table, e.g., virtual-to-physical translation table 532.

5 Virtual-to-physical translation table 532 may be a platform-specific set of translation tables and having entries with virtual memory addresses V and corresponding physical memory addresses P, thereby allowing the memory system to look up a virtual address and find the physical page containing the address. In one embodiment, device driver 214 sets up translation table 532 in MMU 530.

10 During normal operation, physical memory 550 may be the device (e.g., framebuffer 208) and during power management, physical memory 550 may be the host memory (e.g., host memory 209). By using virtual memory, the mapping may be limited to just one or a few pages of memory rather than the size of physical framebuffer 208.

In the alternative, memory space of the size of the physical framebuffer 208 may be allocated in hardware system host memory 209. In one embodiment, the content of the virtual framebuffer may mirror that of the physical framebuffer if data was not redirected.

Framebuffer 208 is an area of memory used to hold, for example, a frame of display data such as is typically used for screen displays. Framebuffer 208 may be a separate memory bank on the display adapter that holds the bitmapped image while the image is being “painted” on the screen. Framebuffer 208 is typically of a size that accommodates the maximum image area on the screen. Framebuffer 208 typically allows application software to access the graphic hardware through a well-defined interface and the framebuffer driver typically presents a generic interface across all software and hardware platforms.

Host memory 209 may be kernel memory of a computer system. Host memory

209 may be configured in accordance with system needs/specification. For example, host memory 209 may be configured to contain a portion of memory (e.g., virtual framebuffer) that is the same size as the actual hardware framebuffer 208. In the alternative, host memory 209 may be configured to include a virtual framebuffer that contains pageable memory. D, data written to and read from hardware framebuffer 208 may be mapped to host memory 209 during the power management period when hardware framebuffer 208 is powered off and unavailable.

Typically, to display a window on a user screen, a window manager incorporated into X Client 222 makes a library call to X Window Server software 220 which then makes a library call to DIX 224 via path 250. DIX 224 makes a call to DDX 226 via path 251. DDX 226 then makes a system call to a device driver 214 (e.g., framebuffer driver) in OS 210 via path 252. Device driver 214 then writes into hardware framebuffer 208 via path 253. Alternatively, DDX 226 may write directly into hardware framebuffer 208 via path 254.

The proposed EStar guidelines require the framebuffer to be powered off after 30 minutes when no keyboard or mouse activities have been detected to reduce power consumption. Powering down the framebuffer during power management may present problems to systems that are running window applications that do not require keyboard or mouse activities. For example, access to the framebuffer when the framebuffer is powered off may cause the system to hang.

In accordance with one embodiment of the present invention, display data is routed to the virtual framebuffer allocated in host memory 209 during the power management mode (e.g., when the framebuffer is powered off). For example, the window manager may make a library call into DIX 224 via path 250. DIX 224 then relays the call to DDX 226 via path 251. DDX 226 then accesses the virtual framebuffer in hardware system host memory 209 in hard drive 207 via path 255. The computer system updates hardware framebuffer 208 after power is restored. Other

communication paths that accomplish the same purpose are possible.

FIG. 3 shows a flowchart of a power management process in accordance with one embodiment of the present invention. FIG. 3 will be described with reference to FIGs. 1 and 2. The process starts in step 400, when X Server 220 is started. In step 402, X Server 220 initializes power management for framebuffer 208 using the policies specified on client machine 200. X Server 220 default values may be used for power management initialization. In the alternative, policies used for the power management initialization may be specified using a graphical user interface tool (GUI) which stores the policies in a configuration file on e.g, a disk. The configuration file may be created and updated using other suitable methods. The configuration file may include information such as, but is not limited to, what devices (e.g., framebuffers, disks, scanners, printers, etc.) are to be power managed; whether the system is to be shut off; if the system is to be shut off, the periods of the day when power management should be considered; description of what constitutes an idle state; and when the system should be automatically powered on after the time management period.

During power management initialization, a DDX 226 power management function pointer is stored by X Server 220 and DDX 226 obtains a mapping from device driver 214 that gives a mapping pointer. Device registers and memory are then accessed by DDX 226 via this mapping pointer.

In step 404, X Server 220 periodically checks to see if client system 200 is in an idle state. Idle state may be determined when there are no external activities at client system 200 for a predetermined period of time. For example, idle state may be determined when no user activities have been detected from the keyboard or the mouse for 30 minutes. Idle state may also be determined by other means, including predefined internal or external conditions, as defined in the configuration file. X Server 220 may keep a clock or a counter, to track time elapsed. X Server 220 resets

the clock or the counter if external activities are detected and continues to check for activity or idleness by repeating step 404. This same technique could be used to detect other conditions that require power management, such as low battery state, for example.

5 If an idle state is detected, DIX 224 in X Server 220 makes a call to DDX 226 via path 251 requesting power management of a certain device (e.g., framebuffer 208) in step 406. In step 408, DDX 226 makes a system call to device driver 214 via path 252 requesting device driver 214 to place the device into power management mode. System calls may be in the form of an IOCTL call, a MMAP call or other suitable
10 processes. IOCTL call is an interface for transporting data and/or control between user code and kernel device drivers while MMAP call is a mapping process.

Device driver 214 allocates a portion in hardware system host memory 209 in
step 410 and re-positions the DDX 226 mapping pointer to point to the allocated host
memory. In this embodiment, the pointer may be returned from device driver 214 to
DDX 226 for error checking. In one embodiment, a new pointer is returned from
device driver 214 to DDX 226 after memory allocation and DDX 226 then uses the
new pointer whenever a device access is made. After memory allocation, instead of
pointing to the addresses that will be mapped to the hardware framebuffer 208, the
mapping pointer now points to the addresses that will be mapped to the allocated
memory in hardware system host memory 209. In other words, the allocated memory
will receive the data intended for the framebuffer.

DDX 226 then makes an IOCTL call to power manager 212 via path 256
requesting power manager 212 to power down the device (e.g., framebuffer 208), in
step 412. In step 414, power manager 212 informs device driver 214 via path 257 that
25 power manager 212 will be powering down the device so that device driver 214 may perform pre-power-down management tasks.

In step 416, device driver 214 performs pre-power-down management tasks

such as saving certain device state so that the device state can be restored after the device is powered back up. Device driver 214 has allocated a portion of kernel memory to store the device state in a software data structure. Any hardware updates which go through device driver 214 are saved in this data structure while the device is powered down. The updates will be written back to the hardware after device is powered back up.

Power manager 212 then sets the device (e.g., framebuffer 208) to the appropriate power level in step 418. In device power management, some devices may have multiple power levels. However, for convenience of this description, it is assumed that when the computer system is in power management mode, the device that is being power managed is powered off.

During the power management period (when hardware framebuffer 208 is powered off), the allocated section of memory acts as a virtual framebuffer where data intended for the actual hardware framebuffer 208 is directed. In other words, data intended for the actual physical framebuffer 208 will be written to and read from the allocated portion in host memory 209 via path 258 during the power off state of the actual hardware framebuffer 208. Since it is assumed that no user is interested in the display during the power management period, and since there is no framebuffer processing of the data written to the device, the data written to the allocated memory may be thrown away. Because there is no data retention requirement, the data may be compressed to a memory area that is smaller than the actual framebuffer memory area. If the data is read, an appropriate value such as a zero or a stored constant value could be read back to allow the process to continue.

During this power management period, server system 200 is still alive and therefore, the kernel is still active. Hence, access to kernel memory or host memory 209 does not cause the system to hang. The applications (e.g., window system software) can continue to operate and update data structures and window system state.

During the off state of framebuffer 208, X Server 220 continues to check for external activities to determine whether to terminate the power management mode in step 420. If there are no conditions that would require the system to power up, framebuffer 208 remains in the power off state.

5 If X Server 220 determines that the device (e.g., framebuffer 208) should be powered up, X Server 220 triggers the wakeup sequence, e.g., to turn the display on. DIX 224 in X Server 220 relays the power up request to DDX 226 in step 422. In step 424, DDX 226 makes a system call to power manager 212 requesting that power be restored to the device (e.g., framebuffer 208). Power manager 212 restores power
10 to the device in step 426. Power manager 212 then informs device driver 214 that power has been restored to the device in step 428. Device driver 214 initializes the hardware to the power up state and restores the device registers from its software data structures in step 430.

In step 432, DDX 226 makes system calls to device driver 214 to leave power management state and redirect the mapping pointers to framebuffer 208. Device driver 214 changes the pointer and releases the allocated memory in step 434. X Server 220 updates the content of the device in step 436.

For a graphic window display, depending on the application, X Server may redraw all of the windows on the screen. X Server 220 is aware of the geometry of all windows on a screen (e.g., number of windows on the screen and their locations, sizes, orders, types and titles). Thus, for an X Server application, X Server 200 may redraw the related window. Where applications outside of X Server 220 provide the contents of the windows, X Server 220 may send a request to the application to redraw the contents of the associated window. For example, X Server may be able to redraw
20 the border of a clock tool window but the clock tool needs to provide the time display.

25 The process returns to step 404 to check for idleness.

While the present invention has been described with reference to particular

figures and embodiments, it should be understood that the description is for illustration only and should not be taken as limiting the scope of the invention. Many changes and modifications may be made to the invention, by one having ordinary skill in the art, without departing from the spirit and scope of the invention. For example, while the invention has been described with reference to a display framebuffer, it is possible to use the same ideas to manage power for other devices.

CLAIMS

We claim:

1. A method for managing power consumed by a computer system, comprising directing access intended for a device coupled to said computer system to an alternate memory space in said computer system when said device is powered off during power management state of said computer system.

2. The method of claim 1, further comprising performing a process that does not require external activities at said computer system to run but accesses said device.

3. The method of claim 1, wherein said directing access comprises mapping data intended for said device to said memory space.

4. The method of claim 5, wherein said directing access comprises performing virtual memory mapping.

5. The method of claim 1, wherein said device comprises a framebuffer.

6. The method of claim 1, wherein said memory space is a portion of a main memory for said computer system.

7. A method for managing power consumption in a computer system, comprising:

placing said computer system in power management mode;
requesting removing power from a device coupled to said computer
system;
allocating a memory in said computer system;
5 removing power from said device; and
directing access intended for said device to said memory while power is
removed from said device.

10 8. The method of claim 7, further comprising executing a process that
includes instructions for accesses to said device, said accesses being directed to said
memory.

9. The method of claim 8, further comprising reading data from said
memory to allow said process to continue running

10. The method of claim 8, further comprising writing data generated from
said process to said memory.

11. The method of claim 7, further comprising detecting an idle state of
said computer system, and wherein said requesting removing power is responsive to
said detection of said idle state.

12. The method of claim 7, further comprising determining whether there
has been external activities at said computer system for a predetermined time.

13. The method of claim 12, wherein said external activities comprise
activities at a keyboard or a mouse coupled to said computer system.

14. The method of claim 7, further comprising:
restoring power to said device;
restoring device state to said device; and
updating said device.

15. The method of claim 14, further comprising releasing said memory and
restoring a first mapping such that data is mapped to said device.

16. The method of claim 14, wherein said updating comprises redrawing
windows on a display device.

17. A method for managing power consumption in a computer system in a
network system having a first computer coupled to a second computer, said second
computer executing a process that accesses a device coupled to said first computer,
comprising:
placing said first computer in a power management state;
allocating range of virtual memory addresses;
removing power from said device; and
directing access from said second computer that is intended for said
device to said memory addresses.

18. The method of claim 17, wherein said process comprises a process that
accesses said device, said process continues running at said first computer.

19. The method of claim 17, wherein said range of virtual memory
addresses correspond to a portion of memory in said first computer.

20. A method for power managing a framebuffer coupled to a computer system, comprising directing access requests intended for said framebuffer to a memory in a computer while said computer system in power management mode and said framebuffer is powered off.

21. The method of claim 20, wherein said framebuffer and said memory each comprises a plurality of addressable locations, and wherein there is a unique address location in said memory corresponding to each address location in said framebuffer.

22. The method of claim 20, wherein said framebuffer and said memory each contains a plurality of addressable locations, and wherein there are fewer addressable locations in said memory than addressable locations in said framebuffer.

23. The method of claim 22, wherein accesses to all addressable locations in said framebuffer are directed to a single addressable location in said memory.

24. A method for managing power consumed by a computer system having a central processing unit, a power management device, and a peripheral device, wherein said power management device controls power to said peripheral device, the method comprising implementing an executable instruction set for directing access intended for said peripheral device to a range of memory addresses when said computer system is in a power management mode and said peripheral device is powered off.

25. A computer system with power management capabilities, comprising a power management circuit capable of directing access intended for a device coupled

to said computer system to a memory in said computer system when said computer system is in a power management mode and said device is powered off.

26. The computer system of claim 25, wherein said power management circuit comprises:

- a server for handling communication between a process and a device;
- a device driver for accessing said device; and
- a power manager for setting power level of said device.

27. A computer system with power management capability, comprising:
a display device;

a framebuffer associated with said display device; and

a virtual framebuffer, wherein access to said framebuffer is directed to said virtual framebuffer when said computer system is in power management mode and said framebuffer is powered off.

28. The computer system of claim 27, wherein said virtual framebuffer contains fewer addressable locations than said framebuffer.

29. A computer readable medium for implementing an instruction set for directing access intended for a device to a memory space during power management mode of a computer system coupled to said device and when said device is powered off.

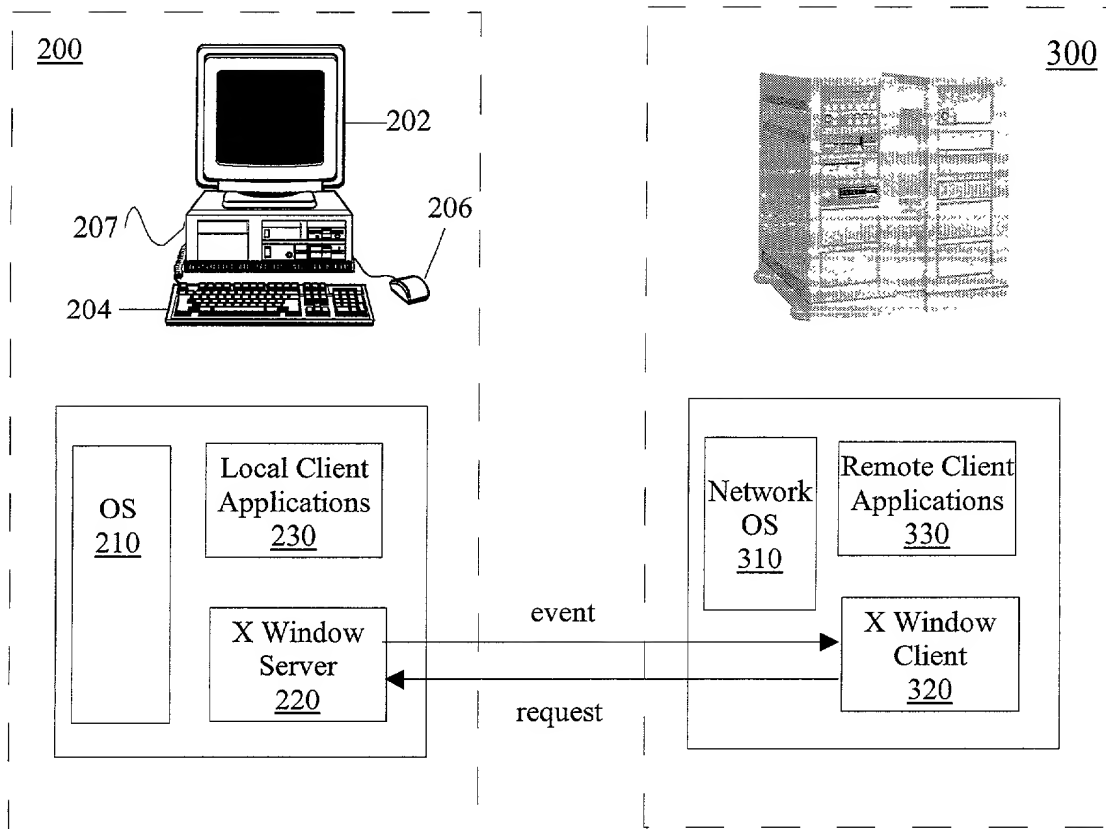


FIG. 1

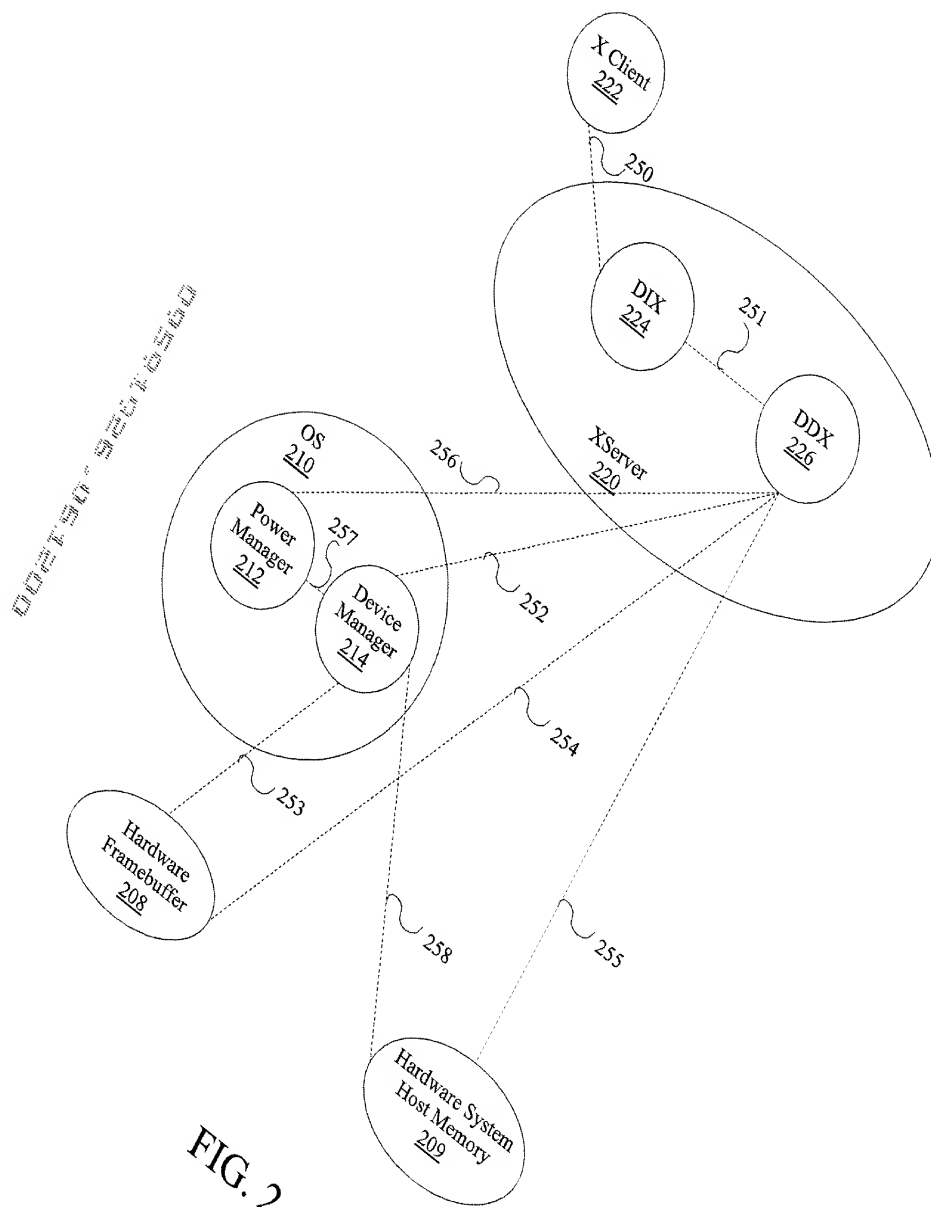


FIG. 2

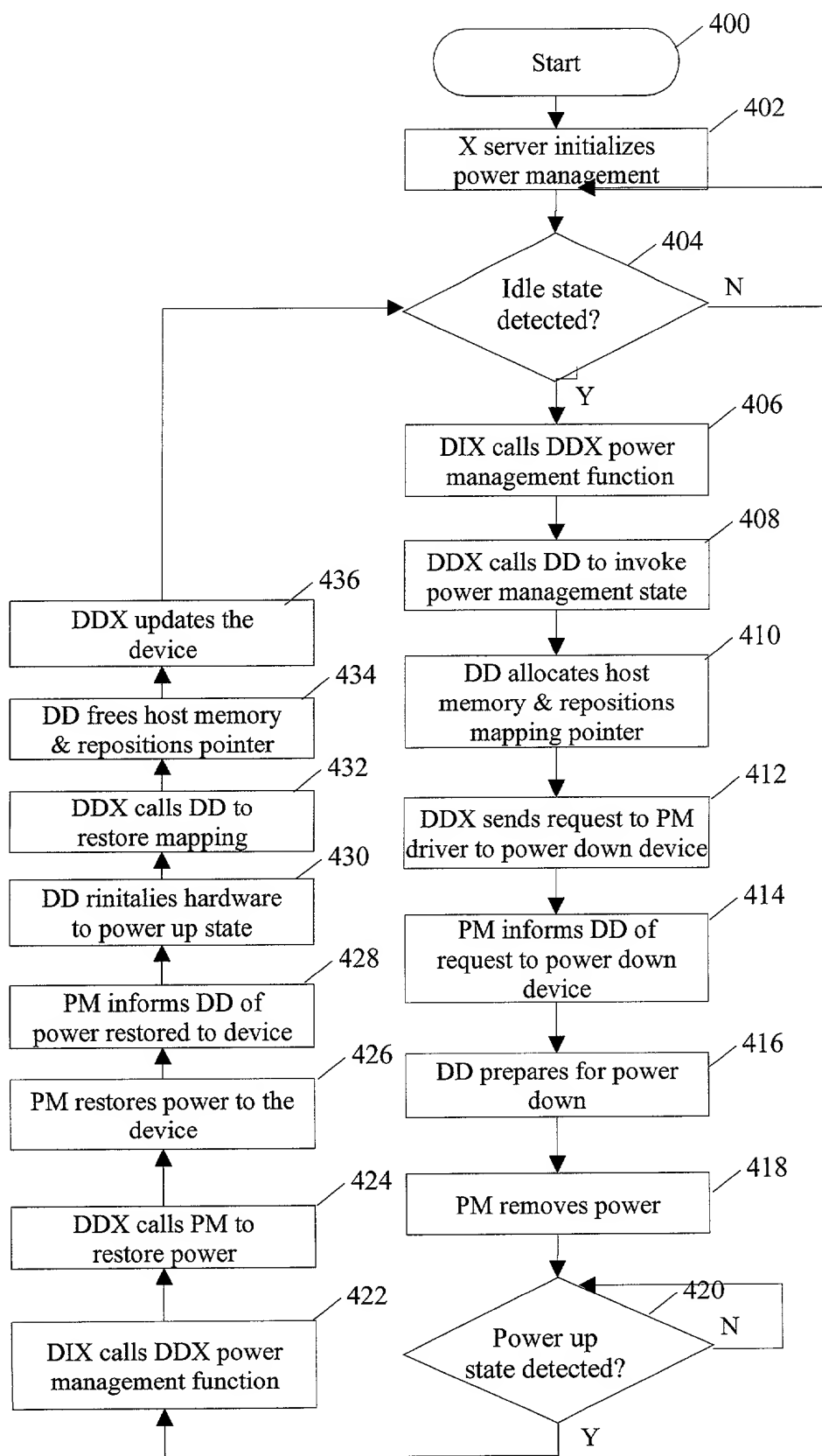


FIG. 3

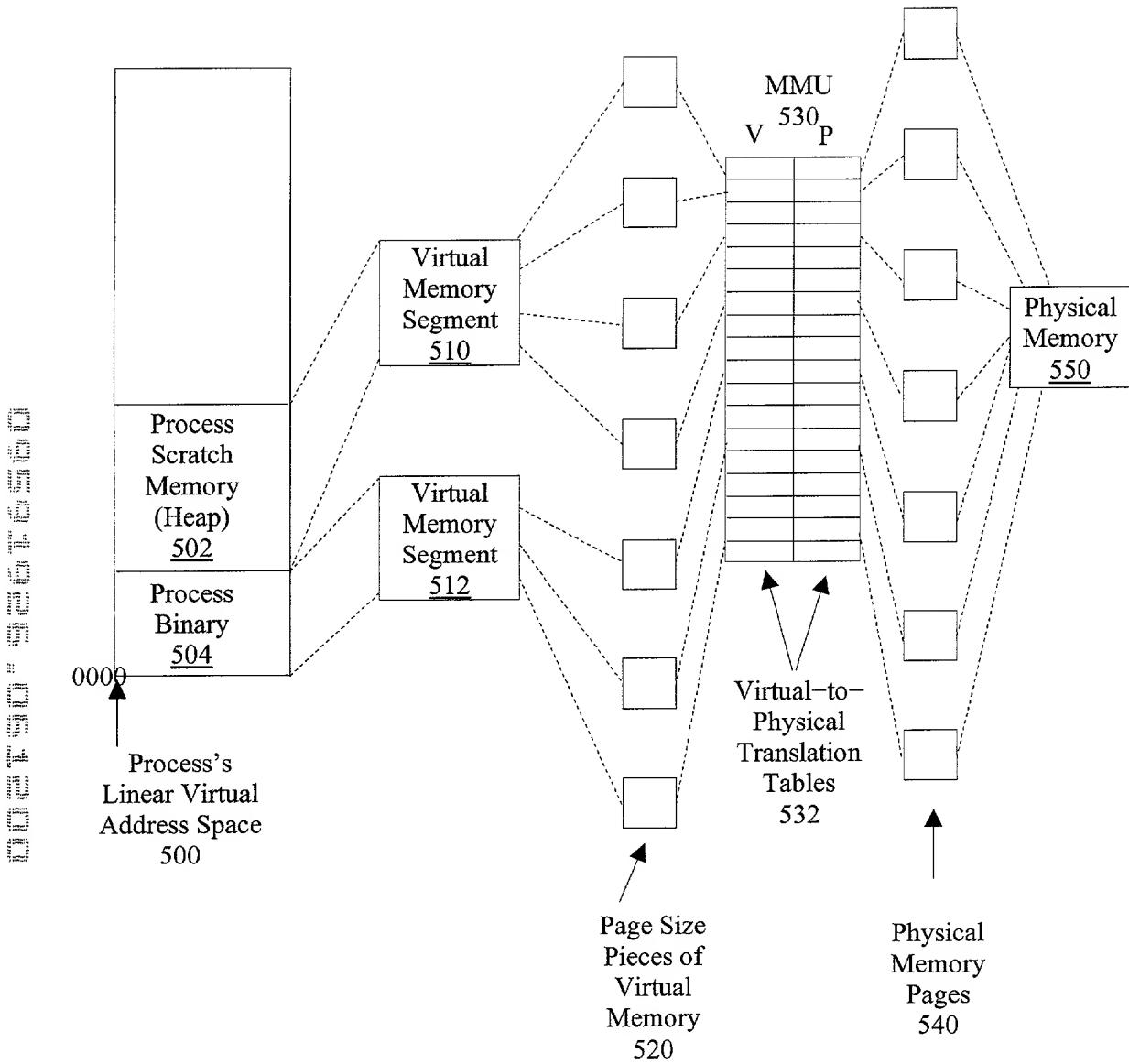


FIG. 4

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

DEVICE POWER MANAGEMENT

the specification of which

☒ is attached hereto.

☐ was filed on _____ as United States Application Number _____
or PCT International Application Number _____ and was amended on
_____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate or of any PCT international application having a filing date before that of the application on which priority is claimed:

<u>Prior Foreign Application(s)</u>			<u>Priority Claimed</u>	
			YES	NO
(Number)	(Country)	(Date Filed)	<input type="checkbox"/>	<input type="checkbox"/>
(Number)	(Country)	(Date Filed)	<input type="checkbox"/>	<input type="checkbox"/>
(Number)	(Country)	(Date Filed)	<input type="checkbox"/>	<input type="checkbox"/>

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional applications listed below

(Application Number)	(Filing Date)
(Application Number)	(Filing Date)

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s), or 365(c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Number)	(Filing Date)	(Status-Patent, Pending, Abandoned)
(Application Number)	(Filing Date)	(Status-Patent, Pending, Abandoned)

POWER OF ATTORNEY

I hereby appoint

Kenneth Olsen, Reg. No. 26,493; Timothy J. Crean, Reg. No. 37,116; Joseph T. FitzGerald, Reg. No. 33,881; Robert S. Hauser, Reg. No. 37,847; Alexander E. Silverman, Reg. No. 37,940, Christine S. Lam, Reg. No. 37,489; Anirna Rakshpal Gupta, Reg. No. 38,275; Sean P. Lewis, Reg. No. 42,798; Michael J. Schallop, Reg. No. 44,319; Bernice B. Chen, Reg. No. 42,403; Kenta Suzue, Reg. No. 45,145; Noreen A. Krall, Reg. No. 39,734; Richard J. Lutton, Jr., Reg. No. 39,756; Monica D. Lee, Reg. No. 40,696 and Marc D. Foodman, Reg. No. 34,110 all of SUN MICROSYSTEMS, INC. with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Address all Correspondence to:

Sun Microsystems, Inc.
901 San Antonio Road, MS UPAL01-521
Palo Alto CA 94303

Attention of Bernice B. Chen Phone: (650) 336-0897; Fax: (650) 336-0530.

SIGNATURES

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application, any patent issued thereon, or any patent to which this declaration is directed.

Full Name of Sole/First Inventor: Carol A. Lavelle

Inventors Signature: _____ Date: _____

Residence: Saratoga, California Citizenship: USA

Post Office Address: 13443 Argonne Drive, Saratoga, California 95070

Full Name of Second/Joint Inventor: Joyce Z. Yu

Inventors Signature: _____ Date: _____

Residence: San Jose, California Citizenship: People's Republic of China

Post Office Address: 4072 Bouquet Park Lane, San Jose, California 95135

Full Name of Third/Joint Inventor: Eric G. Sultan

Inventors Signature: _____ Date: _____

Residence: San Jose, California Citizenship: USA

Post Office Address: 1542 Grace Avenue, San Jose, California 95125